Voice Callback and URS

Universal Routing 8.1.4

1/5/2022

Table of Contents

URS VCB - White Paper

Preface

Universal Routing Server (URS) allows for two types of Voice Callback (VCB) solutions.

Preemptive Agent Reservation

In this process, an agent is reserved in advance before a VCB call is made to reach the customer. This call flow is not that different from inbound interaction routing. An agent is selected for a call, this could be a VCB call, and the customer and the agent are subsequently connected. The connecting phase could be more complex for VCB calls, as the connection entails dialling the customer first as compared to regular routing.

- **Pro**: An advantage of this approach is that the customer need not wait for an agent after answering the VCB call, as the associated agent is already allocated.
- **Con**: Agents can lose time. In this scenario, agents are made to wait for the interaction to arrive during the time the customer is being contacted. If the customer does not answer, the agent wait time is lost. This in turn will affect Agent Productivity Statistics.

Dialing Notifications

The second type of VCB solution is when an agent is not assigned to the call until the customer answers the VCB call. In this case, the customer is contacted in advance, before an agent is allocated to the VCB call. Agent allocation will only be done after the customer answers the call.

- **Pro**: Agents do not lose time as they don't have to wait for the call to be answered.
- **Con**: In an under staffed environment, there could be situations where it might not be possible to route the call to an agent right away and the customer may need to wait in queue.

It is generally assumed that URS uses this second type of VCB solution, the dialing notifications approach where URS can generate VCB notifications under certain conditions.

> **Important**
> This white paper focuses mainly on URS Dialing Notifications.

The purpose of VCB notifications is to facilitate:

- The maximum possible rate of the dialing of outbound calls.
- The minimum possible customer waiting time after answering the call (ideally the customer will be

connected with the first agent who becomes available).

- Beginning with version 8.1.400.40, reduce cases as much as possible, where lower priority calls seep through the not yet dialed/answered VCB calls queue.

The Dialing Notifications VCB solution intends to achieve the following rate of dialing outbound calls: `rate agents become available minus rate of inbound calls + (from version 8.1.400.40) rate of customers not answering their outbound calls.` All references to the optimal or ideal rate in this white paper are based on this.

Another way to look at these 2 types of VCB might be the following:

Consider that for a VCB call the entire process starting from initiation of dialing of outbound call to the customer, waiting for the call to be answered, and then connecting him with the agent, is one routing step. Compare this with a regular call's routing process, which is just connecting the call with an available agent. Routing regular calls is just a small portion of the routing process followed for VCB calls.

In the first type of VCB solution mentioned above, when routing step is started the agent is locked and assigned to this call. When and if the customer answers, he will be connected with this waiting agent. No other calls will be distributed to the agent while he is waiting.

In the second type of VCB solution mentioned above, when the routing step is started the agent is not locked after the VCB call is initiated and can answer any other inbound calls routed to him or other VCB calls (already answered by customers). When and if this current VCB call's customer answers the call, he will be connected with another available agent who happens to be ready on or after the customer answers this VCB call.

VCB Implementation

Standard URS Behavior

In a call surplus Scenario,

1. Agent becomes ready.
2. An *agent becomes ready* event is received from Stat Server.
3. URS builds a queue of all calls waiting for this agent based on priority, waiting time, etc.
4. URS goes through this queue in search of the first *routable* call (starting from the very first call until either it finds such a call or the end of the queue is reached).
5. Once (if) such a call is found, it is routed to that agent.
6. Processing of the *agent becomes ready* event is complete.

> **Important**
> A call is routable if nothing prevents the call from being routed to the agent right now – all thresholds satisfied, all criteria fulfilled, and call is in proper state (that is, selectable, no route delay, etc.).

Step 1: Adding VCB, First Implementation (8.1.400.07)

The first implementation is built on top of the standard URS behavior. Extra functionality is quite simple. While going through the queue of calls waiting for the ready agent and looking for first routable one, URS also looks for the first **VCB notifiable** call. If found, such a call is remembered. By definition, it will have a higher priority than all other **VCB notifiable** calls as well as routable calls, because the call is in the queue ahead of all.

As result, the only difference compared to the standard URS behavior – after handling the *agent becomes ready* event – the **VCB notifiable** call may be identified. And if such a call is identified, a VCB notification will be sent for it.

> **Important**
> A call is VCB notifiable if, the call is in the `DoNotSelect` mode (so it is automatically not routable); the call has the `notifyurl` property defined in its extensions; and, there are no pending VCB notifications for this call.

Reasoning: Every time some agent becomes ready and there is a VCB notifiable call waiting for that agent at the top of queue, URS will send a VCB notification for this call. If the corresponding customer is successfully dialed and the call's `DoNotSelect` mode is removed, then this call automatically becomes first in the queue and routable, so it will be answered when the next agent for this queue of calls becomes ready. This is expected to happen in approximately this queue's **Average Handling Time** (average agent's AHT/number of agents).

Important

If the **dial out hit rate** for the call is set to less than 100%, then URS might look for more VCB notifiable calls (up to 3). The **dial out rate** is set with the `DialOutSuccessRate` [rate of success in percent] strategy function . *0*, if used will be considered as *100%*. After finding the next VCB notifiable call, URS will use the provided value to consider as a probability that the customer answers the call. And URS will continue to search for more VCB notifiable calls with the probability % (100 - provided value). However, no more than 3 calls can be selected in total. For example, a DialOutSuccessRate of 70% will result in URS sending out one VCB notification with a probablity of 0.7%, two VCB notifications with a probability of 0.21% (03*0.7), and three VCB notifications with a probability of 0.09% (that is, 1.39% on an average).

Below is a listing of DialOutSuccessRates and their corresponding average number of VCB notifications that the `agent ready` event might generate:

DialOutSuccessRates	Avg. No. of VCB Notifications
100%	1
90%	1.11
80%	1.24
75%	1.3125
70%	1.39
60%	1.56
50%	1.75
40%	1.96
30%	2.19
20%	2.44
10%	2.71

Important

In the following sections (for simplicity) it is assumed that we have a 100% dial out hit rate (so that URS will look for only one VCB notifiable call each time).

Step 2: Improving VCB Notification Rate (since 8.1.400.23)

As can be seen, the functionality in Step 1 does (almost) not change from the standard URS behavior on processing the **agent becomes ready** event. This simplicity however might affect the rate of VCB notifications, which is about two times less than an optimal rate. So, while VCB customers are connected quickly with agents, the queue of VCB calls might be handled slowly.

The basic use case considered in this step - If only VCB calls are present in the system (that is, all calls that URS has are VCB ones and there are no other call types) and hit a dial-out rate of 100% - then the expectation is that the rate of VCB notifications will be the same as the rate of the **agent becomes ready** events (as there are no inbound calls occupying agents). This, however, does not happen within Step 1.

When two **agent becomes ready** events activate only one VCB notification:

- The **Agent 1 becomes ready** event triggers a VCB notification for Call 1.
- Call 1 is connected with a customer and the **DoNotSelect** mode for this call is cleared (so it is no longer VCB notifiable and is first in the queue).
- The **Agent 2 becomes ready** event triggers Call 1 to be sent to Agent 2, but it does not trigger a VCB notification as there are no VCB notifiable calls ahead of Call 1.
- Only when Agent 3 is ready will the next VCB notification be sent for the next VCB call.

To handle the above use case, the standard URS behavior on processing **the agent becomes ready** event was extended and goes beyond what is needed for simply routing calls. Specifically, URS does not stop when a routable call is found, but will try to continue exploring the queue of calls in search of a VCB notifiable one (if such a call is not already found). Previously, in URS, there was no such concept of *queue exploration* after a routable call was found. This behavior is limited to an extent so as to not affect URS performance in standard but not VCB-related cases.

The main idea behind this step is that, if a call is routed to an agent and the next call after it is a VCB notifiable one, then it is alright to send a VCB notification.

Implementation: URS continues to go through the queue of calls after a routable call is found until one of the below happens (first):

- A VCB notifiable call is found.
- Another routable call is found (meaning, when the next agent becomes ready, this call will be routed to that agent). All other VCB calls, even if they exist, have a lower priority and will not be routed. No more than 20 calls are checked (this is configurable, and is 20 by default).
- No more than 20 calls are checked (this is configurable, and is 20 by default).

Step 3: Further Improving (*increasing*) VCB Notification Rate (since 8.1.400.26)

While Step 2 is expected to help achieve a VCB notifications rate that is closer to the ideal rate, there are still use cases where this is not enough. As a result, extra logic is embedded from version

8.1.400.26.

The implementation described in Step 2 is updated as follows:

Use Case A

URS continues to go through its queue of calls after a routable call is found until one of the following happens first: - A VCB notifiable call is found. - Another routable call (but not a former VCB one) is found (no more than 20 calls is checked, this number is configurable).

Compared to Step 2, here URS continues to go through the queue of calls after a routable call is found until one of the following happens first:

- A VCB notifiable call is found
- Another routable call (but not a former VCB one) is found
- 20 calls are checked (this value is configurable and 20 by default)

The point here is taking into account the nature of another routable call. If it happens to be a former VCB call (a call that becomes routable due to a VCB notification that URS sent previously), then it should not prevent a search for other VCB calls. This use case can become significant on high call rates/VCB notification rates when more than one VCB call can become routable more or less simultaneously.

Use Case B

While the following use case could be an extremely rare one, it is still worth addressing.

If during the processing of an **agent becomes ready** event, no routable call was found, then arises a situation where there is an available agent without any call and it is expected that one more agent will become available soon (in queue AHT time). In such cases, URS will send two VCB notifications (meaning, a single **agent becomes ready** event can sometimes trigger two VCB notifications) one for the current agent and a second one for the agent who will become available next.

Since 8.1.400.26, this functionality is controlled with the **n10** parameter of the VCB option (see VCB Configuration) and is disabled by default.

> **Important**
>
> Before 8.1.400.40, this functionality was also disabled in a multi-URS environment (n4=1, see VCB Configuration).

Use Case C

URS has no control on how many VCB calls it gets and the queue of VCB calls could still grow (for example, if there are not enough agents). URS considers this as a normal situation if VCB calls and inbound calls are still treated fairly. A potential issue here is that a VCB notification does not block the agent and a lower priority inbound call will be routed to that agent (meaning, the low priority call will be routed ahead of higher priority VCB calls as they cannot be routed yet). In other words, low priority inbound calls might seep through the waiting queue of calls. To avoid this scenario, the following optional logic is added:

If for whatever reasons, URS gets a significant surplus of high priority VCB calls, then it tries to compensate for it. A surplus of high priority VCB calls occurs in the following condition:

- While going through a queue of calls upon processing **an agent becomes ready** event (in search of the first routable call)if URS finds that the head of the queue contains a large (>25, adjustable) amount of not-yet-tried VCB calls. In other words, there are a lot of high priority VCB calls and not a single high priority *non-VCB* call.

If URS detects such a situation, it stops looking for routable calls for this agent so no call is routed to the agent upon processing the **agent becomes ready** event. Referring to case B above, this means that two VCB notifications will automatically be sent for this **agent becomes ready** event. Also, the agent is marked as reserved for processing VCB calls only (or high priority inbounds calls). Such reservation continues while the surplus of high priority VCB calls exists.

> **Important**
>
> Before 8.1.400.40, this functionality was disabled in a multi-URS environment (n4=1, see VCB Configuration).

Step 4: Further Improving (*correcting*) VCB Notification Rate (8.1.400.26)

While looking for routable calls (applicable for all steps), URS checks not only the call state but the target (agent) state too. Specifically the following checks are carried out if the agent is not blocked completely:

- Is call state OK?
- Is agent valid?
- Is agent blocked?
- And for every agent's DN: Is DN valid? Is DN blocked?

It is possible that a call is considered not routable only because of agent/DN blocking. In such cases, URS can potentially pass over a big number of calls as cannot be routed only because of agent blocking and reach a VCB notifiable call located well behind in the queue. Triggering a VCB notification in such a case could be incorrect – as another (or the same) agent becomes available the specific VCB call will not be routed to the agent (as there are a lot of other routable calls waiting).

In general, correctly distinguishing the above described case is a high CPU usage activity (because an agent can be partially blocked, that is, blocked for one call and not blocked for others). As a simplified alternative the following is done:

The cycle described in Step 1 is modified as follows: While going through the queue of calls waiting for a ready agent and looking for the first routable call, URS also looks for the first VCB notifiable call. However, as soon as such a call is not routable because of agent/DN blocking, then, since that moment the search for a VCB notifiable call is cancelled. In other words, blocked agents/DNs can trigger VCB notification only if a VCB call is the first call among all calls waiting for that particular

agent).

Improvement described in Use Case B of Step 3 should also be disabled if agent is blocked (as this agent is not available and there is no justification to send the agent a second notification).

Improvement described in Use Case C of Step 3 should also be disabled if agent is blocked (as there is no way to check presence of routable calls in such a case).

Step 5: Further Improving (*extending*) VCB Notification Rate (8.1.400.36)

A multi-URS VCB environment is extended to allow using of an adjustable dial out hit rate. Previously dial out hit rates in case of multiple URSes was forced to always be 100%.

However, there are some limitations though (See Multiple URSes) and as a result it is advised to increase the priority of a VCB call after an outbound call was answered by the customer.

Step 6: Further Improving (*adjusting*) VCB Notification Rate (8.1.400.37)

The functionality implemented for Use Case B in Step 3 is disabled by default and explicitly controlled with the VCB option (n10=1, see VCB Configuration).

Step 7: Further Adjusting VCB Notifications (8.1.400.40)

From version 8.1.400.40, another requirement was being considered; prevent, as much as possible, lower priority inbound calls from seeping through the queue of not yet dialed/answered VCB calls. Basically, the only absolutely reliable way to achieve it is through VCB Type 1 (blocking/reserving agent after a VCB call). The below modifications were discussed in an attempt to address the issue within VCB Type 2 (notification based).

A way of addressing this requirement, is to provide a big enough pool of VCB calls already answered by customers, so that low priority inbound calls will not have chance to squeeze through the queue. Though seeping of low priority calls might only happen at the moment an agent is ready and there are no answered (ready for routing) VCB calls, a big pool of answered VCB calls would mean that these customers might wait for quite a long time before they can be connected with an agent - as the main reason for dialing them was to block inbound calls and not to facilitate a minimum waiting time after answering.

In effect, this requires URS to maintain a balance between *under-dialing* (some portion of inbound calls seeps through the queue of VCB calls) and *over-dialing* (once answered, customer waits too long before being actually connected with an agent) conditions. We can achieve this as follows:

a. To facilitate a situation where, URS always has VCB interactions that are already answered by customers, then URS must work in a slightly *over-dial* mode. The current default *optimal* rate (one notification per one agent ready event) will never be sufficient. The default rate of notifications URS will use now is 1.5 times higher than the rate at which agents become ready. One agent ready event on an average triggers 1.5 notifications (it is slightly less than the rate that resulted from a `DialOutSuccessRate` of 60). This can be controlled with the `n11` parameter of the VCB configuration option (see VCB Configuration). However, this parameter will be ignored if the `DialOutSuccessRate` function is used by a VCB solution.

b. The inherent *over-dialing* must be controlled and suspended/reduced when URS detects that customers of connected VCB calls are beginning to wait too long, as the criteria that URS uses is the time that connected VCB calls are already waiting for and also the number of already connected VCB calls. Specifically, when URS detects that a customer is waiting for more then one minute, it assumes a *over-dial* situation (controlled with the `n12` parameter of the `vcb` configuration option). The same is also true, if URS detects that there are already 5 answered VCB calls (controlled with the `n13` parameter of the `vcb` configuration option). In an *over-dial* situation, by default URS stops generating new VCB notifications while the *over-dial* situation continues.

With the `n14` parameter of the `vcb` configuration option, URS can be configured to still consider making VCB notifications but with a much smaller rate.

> **Important**
>
> This version changes the default URS VCB behavior. As in, it has a much higher dial out rate. This factor should be considered, when upgrading to 8.1.400.40. If there is a necessity to preserve the previous VCB behavior, then either the `n11` parameter of the `vcb` configuration option must be explicitly set to 100 or the `DialOutSuccessRate` function must be used. It is strongly advised not to mix both the approaches to control the dial out rate (`vcb:n11` parameter and `DialOutSuccessRate` function). If the `DialOutSuccessRate` function is going to be used, then it should be executed for every VCB call.

Note: The *over-dial* protection if enabled (with the `n12` and/or `n13` parameter) works irrespective of whether *over-dialing* results from the default dial out rate or using the `DialOutSuccessRate` function.

c. Using the jump priority functionality (the `n9` parameter of the VCB configuration option) might result in a sharp slow down in the VCB notifications rate. To handle this situation, when looking for VCB notifiable calls after a routable call is found (see the `n5` parameter of the `vcb` option in VCB Configuration), URS will not count the dialed but not answered VCB calls towards the `vcb:n5` parameter.

d. At times, there might be a few queues of VCB calls which are understaffed and result in extremely high wait times for processing every single call (20 to 30 minutes or even more). Such cases of very slow queues generally do not fit the notification model described here as it is practically not possible to find a right moment for sending the notification. The customer will almost always have to wait a significant amount of time after answering an outbound call. Cases like this are better addressed with the first type of VCB solution where the agent is actually allocated to an interaction from the moment a VCB call is dialed to a customer. As a compromise though, it is possible to configure a *hybrid* approach - in some cases URS might actually reserve an agent for a VCB call at the same moment that a dial out notification is sent and keeps the agent reserved until the customer answers. As this is effectively VCB Type 1 solution within a VCB Type 2 solution (see Preface section) it results in an

instant connection of the call/customer with the agent, but the agent can still loose a part of his productive time waiting. This waiting might become bigger (up to the value in the n2 parameter of the vcb configuration option in VCB Configuration) if the customer does not answer and also in cases of a multi-URS VCB. This functionality is controlled with the n15 parameter of the vcb configuration option in VCB Configuration. It specifies a minimum AHT for the queue where a VCB call is located and when this time is exceeded URS switches to the agent reserving mode (for calls from this queue only). By default, the value of this parameter is 0, meaning that this functionality is disabled.

In addition to checking queue AHT, URS might also check queue MHT (maximum handling time) and whether it exceeds the value of the n15 parameter. Checking is done within the timeframe (seconds) defined in the n16 parameter, which is 0 by default (meaning that the functionality is disabled). If within the defined timeframe, at least one call handling time for the queue exceeds the value specified in the n15 parameter, URS will switch to the agent reserving mode (for calls from this queue only).

> ### Important
>
> Thoroughly consider all pros and cons if enabling this *slow queue* functionality. A major use case for this functionality is that, if there are only 2 or 3 agents to handle the calls from a queue and handling one call takes around an hour, then, maybe to facilitate an acceptable wait time (after customer answers an outbound call) it is alright to sacrifice one minute of the agent time. If so, then the vcb:n15 parameter can be set to around 1200 (20 minutes) and URS begins to reserve the agent after the in-progress VCB call, if the next available agent is estimated to appear only after 1200 or more seconds.

Considering all the above, the process that URS follows (on a relatively low level) when going through a queue of calls waiting for an agent (when the agent becomes available) is as follows:

Here,

- fresh_vcb_counter is the number of VCB calls not yet dialed even once
- vcb_main_pool_size is the number of available slots in the pool of stored data for dialing VCB calls
- vcb_extra_pool_size is the number of available slots in the extra pool (one more VCB call might be stored for accelerated notification if no routing takes place or if the first and only VCB call from the main pool is not answered)
- vcb_after_route_total - number of calls in queue after a call was already routed to an agent
- vcb_after_route_former_vcb - number of calls in queue after a call was routed (that is, calls which are routable now and were former VCB calls, which had customers waiting)

Agent becomes ready.

- fresh_vcb_counter = 0
- vcb_main_pool_size = 1 to 3 (depending on how many notifications URS will generate on behalf of the agent)
- vcb_extra_pool_size = 1

If current ratio for VCB notifications is different from 100% (1:1), vcb_extra_pool_size = 0 (if

vcb_main_pool_size might be more than 1, then the extra pool is not used).

If agent ID, place, or DN is blocked (or was used in recent past), vcb_extra_pool_size = 0 (see Step 4 above).

Loop 1:

For every call waiting for this agent (in priority/best fit/timing order):

- if call is routable, try to route it.
  - if success (call is routed), exit from Loop 1.
  - if failure (call is not routed) because agent/place/DN is blocked, vcb_main_pool_size = 0 (see Step 4 above).
- if call is VCB notifiable:
  - if call was not yet dialed, fresh_vcb_counter++
  - if fresh_vcb_counter > vcb:n8, exit from Loop 1 (if at this point, both pools of VCB calls are completed, vcb_main_pool_size and vcb_extra_pool_size are 0)
  - if vcb_main_pool_size > 0, store the call in the main VCB pool, and vcb_main_pool_size-- else, if vcb_extra_pool_size > 0, store the call in the extra VCB pool and vcb_extra_pool_size--

At this point, if a call is routed and there are not enough stored VCB calls (vcb_main_pool_size > 0 or vcb_extra_pool_size > 0), then Loop 1 is exited and Loop 2 comes into effect.

Loop 2:

- vcb_after_route_total = 0.
- vcb_after_route_former_vcb = 0.

For every call waiting for this agent, after the just routed call (in priority/best fit/timing order):

- if call is not a dialed but not answered VCB call, vcb_after_route_total++
- if vcb_after_route_total > vcb:n5, exit Loop 2
- if call is routable:
  - if call is not a former VCB call (that is, call is a regular inbound call) then exit Loop 2
  - if call is a former VCB call, then vcb_after_route_former_vcb++
  - if vcb_after_route_former_vcb > vcb:n13, then enter the *overdial* mode
  - if the customer for this call has already been waiting for a considerable time, then enter the overdial mode
  - if overdial mode,
    - with probability of 100 - vcb:n14, then empty pools of collected VCB calls (if any) and exit Loop 2
    - if pool of collected VCB calls is not empty, then leave one call in the queue, and exit Loop 2
    - vcb_main_pool_size = 1 (with probability vcb:n14, allow room just for one VCB call)

        - `vcb_extra_pool_size= 0`
    - if VCB call is notifiable:
        - if `vcb_main_pool_size > 0`, store the call in the main VCB pool and `vcb_main_pool_size--`
        - else, if `vcb_extra_pool_size > 0`, store the call in the extra VCB pool and `vcb_extra_pool_size--`
        - if `vcb_main_pool_size = 0` and `vcb_extra_pool_size = 0`, then exit Loop 2 (there are enough stored VCB calls)

At this point, both Loop 1 and Loop 2 are exited.

- if there are calls in the main VCB pool, send VCB notifications for each one of the calls
- if there are calls in the extra pool,
    - if no call was routed and `vcb:n10` is `1`, then send a VCB notification for this call too.
    - otherwise store information about this call in call1 from the main VCB pool (could be only one call there too) - if call1 customer does not answer, the VCB notification will be sent for this stored call

Additional VCB features in this version are:

- Option `vcb` (see VCB Configuration) can be set at the call level with the `SetCallOption['vcb', '...']` function. Not every parameter of this option has a meaning at the individual call level and accordingly only a few parameters of this option can be set at the call level. Specifically, the `n1`, `n2`, `n6`, `n9`, `n17` parameters can be set the call level. Use values -1 for all other parameters. If the value of a parameter is set to -1, its value will be taken from vcb option at the URS level.
- The `vcb` option is dynamic. It does not require a URS restart if modified.
- Every parameter of the `vcb` option get its own name and can be specified on its own (see VCB Configuration).
- The default value of the `n8` parameter of the `vcb` option is 25 (previously, it was 50).
- The default value of the `n1` parameter of the `vcb` option is 60 (previously, it was 30).

VCB Related Logging in URS

While going through a queue of calls in search of the first VCB notifiable one for which a notification can be sent, URS logs a short disposition code in the log about every checked VCB notifiable call. The corresponding logging message has the following format:

```
_M_I_connid [10:21] call VCB states N (d1:d2) (d3:d4) text (d5 d6)
```

For example:

```
_M_I_036f02849e162002 [10:21] call vcb states 1(0:0) (0:0) (1376 0)
```

- The **d1**, **d2**, **d3**, and **d4** parameters are specific to the major VCB status (**N**) and used for its clarification.
- The text message parameter is usually empty and used to report specific circumstances like over-dialing.
- The **d5** parameter denotes the duration that the VCB call is in processing.
- The **d6** parameter denotes the number of notification attempts for this call so far.

A message is printed for every checked VCB call (that is, every call in **DoNotSelect** mode and having `notifyurl` defined).

- N = **0**, if for this call the VCB notification was already sent recently, within n2 seconds (**d1** - how much time remains until n2(+n17) expires, **d2** - 0, **d3** - dial out success for the call, **d4** - accumulative dial out success rate). See VCB Configuration for information on the n2 parameter.
- N = **1**, if call cannot be routed right now (even if there was no **DoNotSelect** mode) due to call status (**d1** and **d2** - 0, **d3** - dial out success for the call, **d4** - accumulative dial out success rate).
- N = **2**, if the VCB notification was already done on behalf of the currently processed target (agent) recently, within n3 seconds (**d1** - how much time remains until n3 expires, **d2**, **d3**, and **d4** - 0). See VCB Configuration for information on the n3 parameter.
- N = **3**, if the call is good for a VCB notification (**d1** - disposition about one more VCB, **d2** - aqt if `vcb:n15` is on, **d3** - dial out success for the call, **d4** - accumulative dial out success rate).
- N = **5**, if the call has already been scheduled (but a VCB notification has not been sent yet) and it is not first among such calls (**d1**, **d2**, **d3**, and **d4** - 0).
- N = **6**, if the first call that has already been scheduled (but a VCB notification has not been sent yet) is used as an extra/alternative VCB call, (**d1**, **d2**, **d3**, and **d4** - 0).
- N = **7**, if the n7 parameter is set to 0 and URS checks for the VCB call extra condition and if it will fail due to *threshold/ready* conditions (**d1** and **d2** - 0, **d3** - dial out success for the call, **d4** - accumulative dial out success rate). See VCB Configuration for information on the n7 parameter.
- N = **8**, if the n7 parameter is set to 0 and URS checks for the VCB call extra condition and if it will fail due to *target/agent validness* verification (**d1** and **d2** - 0, **d3** - dial out success for the call, **d4** - accumulative dial out success rate). See VCB Configuration for information on the n7 parameter.

- N = **9**, if the n7 parameter is set to 0 and URS checks for the VCB call extra condition and if it will fail due to *target/agent DN validness* verification, (**d1** and **d2** - 0, **d3** - dial out success for the call, **d4** - accumulative dial out success rate). See VCB Configuration for information on the n7 parameter.

- N = **10**, if the next good VCB call after the one that is already selected is used as an extra/alternative VCB call (**d1** - 0, **d2** - aqt if vcb:n15 is on, **d3** - dial out success for the call, **d4** - accumulative dial out success rate).

As an exception, when URS goes through a queue of calls after a routable call is found, this message might also be printed for calls that are routable (and as a result not in the **DoNotSelect** mode).

- N = **-1**, if the call is not a VCB call at all; URS should not try to look further for a VCB call after this (**d1**, **d2**, **d3**, and **d4** - 0). See Step 3 (Use Case A) in VCB Implementation.

- N = **4**, if the call is a former VCB call; URS should continue to look further a VCB call after this (**d1** - time live customer waits, **d2** - number of former VCB calls so far, **d3** - vcb:n12, **d4** - vcb:n13). See Step 3 (Use Case A) in VCB Implementation.

When URS selects a VCB call for notification and starts the notification process, it logs a message similar to the following:

If the notification is actually sent (here n2 is the expected AHT if it is counted, n3 – timestamp (UTC) when VCB notification for this call can be repeated, n4 – number of seconds left to this moment of time, n5 – counter of *failed* dial notifications for this call, pvq – points to internal queue containing the agent):

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web notification <http://10.179.117.52:8080/genesys/
1/ors/scxml/... > sent (hints: 0 n2 n3 n4 n5 pvq)
```

If the notification is delayed due to high AHT or necessity to send reserving request (here n1 is 1 if current notification is delayed due to high EWT (here, n2 = 1 if reserving request is sent, n3 – timestamp when the VCB notification for this call can be repeated, n4 – in case of high EWT – for how long notification is delayed, pvq – points to internal queue containing the agent):

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web notification <http://10.179.117.52:8080/genesys/
1/ors/scxml/... > delayed (hints: n1 n2 n3 n4 0 pvq)
```

If it is an attempt to send one more notification for the call which already has a delayed notification, then the notification will not be sent and URS records this message in the log (here, n1 is 1 if current notification is delayed due to high EWT, n2 = 1 if current notification is delayed due to agent reserving request, n3 – timestamp when the VCB notification for this call can be repeated, n4 – number of seconds left to this moment of time, pvq – points to internal queue containing the agent):

```
16:31:12.006_A_E_036f02849e162002 [0E:22] web notification failed (hints: n1, n2, n3, n4, 0,
pvq)
```

If the notification was sent but resulted in no response and URS is going to send another notification, the following will be printed in relation to the *expired* notification (n3 – timestamp when VCB notification for this call can be repeated, n4 – number of seconds left to this moment of time, timed out is used if provided timestamp is actually expired, cleared if doing it before the timestamp expired):

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web notification cleared|timed out (hints: 0, 0,
n3, n4, 0, 0)
```

If any notifications sent for this call is aborted (due to a high number of previously sent unanswered

notifications (in hints URS places the number of unanswered notifications):

```
<tt>16:31:12.006_A_E_036f02849e162002 [0E:22] web notification terminated (hints: 0 0 0 0 n
0)</tt>
```

When URS sends a VCB notification because of update of some target (agent) this target is blocked for some time to prevent/provoke sending multiple notifications (see n3 in VCB Configuration). The appropriate logging message looks like:

```
16:31:12.006_M_I_036f02849e162002 [0E:25] SO(2ac076be44e8 13 2) ten=Resources
name=1805@STAT01.A: vcb notification timeout set: 1464877932 (+60)
```

URS can also clear such target blocking of VCB notifications if needed with an appropriate message as follows:

```
16:30:21.745_M_I_0000000000000000 [0E:25] SO(2ac076be44e8 13 2) ten=Resources
name=1805@STAT01.A: vcb notification timeout clear: 0 (+0)
```

VCB Configuration

URS has an option `vcb`, which controls the different aspects of the VCB notification functionality in URS.

> **Important**
>
> The `vcb` option is dynamic from URS version 8.1.400.39.

The value of this option is in the format of a sequence of numbers: `n1:n2:n3:n4:n5:n6:n7:n8:n9:n10:n11:n12:n13:n14:n15:n16:n17`, and by default is, `60:90:90:0:20:0:1:25:0:0:150:60:5:0:0:0:300`. The `vcb` option is set at the URS level (though it is possible to adjust it at a call level, see **Step 7** in VCB Implementation).

The meaning of these numbers is as follows:

- `n1` (**AHT threshold**, by default (from 8.1.400.40) 60, before that 30) – when an agent ready event triggers a VCB notification, URS can send the notification immediately or with some delay. If expected time when next agent becomes ready (queue AHT time) is high, then URS can decide to delay the notification to not alert the customer too early. Specifically, if this time (AHT) is more than `n1` seconds then URS delays notification by AHT/2 seconds. AHT time here is counted for the router's queue (that the VCB call is in). It is the same value as returned by the URS function `RvqData[internal_queue_id, RVQ_DATA_AHT]`.

- `n2` (**block call**, by default 90) – logically can be interpreted as the maximum time needed to connect an outbound call with the customer. When the VCB notification for a call is sent, URS might not know what happens with this notification - was it processed somehow, what is the result of this processing (there is no formal feedback event). As a result, the VCB call might remain in the VCB notifiable state and trigger another notification on behalf of another agent, and so on. To prevent a single VCB call from *hogging* all agents once the VCB notification is sent for a call, the call is blocked from sending any other VCB notification for the next `n2` seconds. If, after expiring of this interval the call is not yet routed and is still in the VCB notifiable state, then the next VCB notification can be applied to this call. The other option affecting call blocking time is `n17` (in effect, call blocking time is `n2+n17`).

- `n3` (**block agent**, by default 90) – effectively the same as `n2`, but in context of agents. If an agent becomes ready and triggers a VCB notification, this agent can remain ready (no other calls sent to him) and as a result trigger other VCB notifications. To prevent this from happening (not to generate too many notifications) this agent is blocked from triggering any other VCB notifications (for other calls) for `n3` seconds.

- `n4` (**multi-URS VCB**, by default 0) – can be 0 or 1 and indicates whether any other URS instance exists that can route a call to the same agents. When the VCB notification for some call is made it is expected that this VCB call is the first in queue. Every URS instance however, knows only its own queue's meaning and other URS instances can have higher priority calls. In such a case, no VCB notifications should be made. In other words, this flag indicates if there are multiple URS instances or not. If the multi-URS mode is set for VCB notifications then it also must be set for regular routing – in other words URS also must have the `agent_reservation` option set to on (if not, this `n4` value will be ignored). If `n4 = 1`, then before making a VCB notification (or before routing) URS will try to explicitly reserve the agent triggering the VCB notifications with priority data taken from the VCB call (or the call that is

routed). VCB notifications will be distributed only if/when URS gets a confirmation on this reserving request. The reserving request is made on behalf of the agent DBID and as a result does not interfere with routing reservation. It also means that when a call is selected and routed it might result in 2 reserving requests to T-Server – one VCB related and another regular routing related.

> **Important**
>
> This flag, if set works even if URS does not participate in VCB. It just causes URS to signal about the call when routing the call. And VCB URSes (if they happen to exist) will use them to compare priorities and adjust the sending of VCB notifications.

Note: The above described agent reservation does not work for SIP cluster nodes (it will not accept reservation on behalf of any object except the actual DN). So, when SIP cluster is part of the deployment, ensure that if this flag is set, the reserving request is not sent to the SIP cluster nodes.

Since 8.1.400.23

- n5 (**after route queue max**, by default 20) – controls how far URS will go through a queue of calls after a routable call is found. See **Step 2** and **Step 3** in VCB Implementation.

Since 8.1.400.26

- n6 (**max notifications**, recommended to set to 5 but for compatibility by default is 0) - can be used for tracking and eliminating dead VCB calls. Lost (stuck) VCB calls are those that do not exist in the module accepting VCB notifications and as a result VCB notifications for such calls are just ignored. This can easily result in a small amount of dead VCB calls clogging the distribution of VCB notifications completely. In this scenario, all VCB notifications (or a major part of them) will be for such dead calls. If n6 is not 0 then URS counts for every VCB call how many unanswered VCB notifications in a row was done. The counter is cleared only if URS gets an external message (it does not matter what the message is about) for this call – such as a web request or a message from ORS. If the counter however, manages to reach n6 then such a call is temporarily removed from the pull of VCB calls for 20 minutes. If, after expiration of this time, the counter is still not cleared, then one more VCB notification for this call is allowed and if it still results in no action, the processing of this VCB call is terminated inside URS.

- n7 (**use any agent**, by default 1) – allows using any ready agent for for triggering a VCB notification. The routing strategy (including those processing VCB calls) can impose many different extra conditions on agents the call can be routed to (different thresholds and so on). If n7=0, then URS checks all those conditions for VCB calls to become eligible for VCB notifications (in a general scenario, it is not possible to predict when the customer will answer the call and some other agent will become ready – whether or not all those conditions are satisfied).

- n8 (**check queue**, by default 25 from 8.1.400.40, before that 50) - controls the definition of the *surplus of high priority VCB calls* case. See **Step 3** (**Use Case C**) in VCB Implementation.

- n9 (**jump priority**, by default 0) – allows to automatically boost call priority by n9 when VCB calls become routable (customer answers the call).

> **Important**

> A consequence of jumping priority to be aware of in versions before 8.1.400.40 is that, answered VCB calls might be concentrated at the beginning of the queues (as having moved up in priority). This means that a search of VCB notifiable calls will effectively be moved to post routing which is limited by the n5 parameter of `vcb` option (by default 20). Taking into account the existence of not answered calls, that might leave very little room for finding the next VCB calls to dial. In other words, jumping priority slows down the notifications rate.

- `n10` (**one blocker**, by default 0) – deprecated, set it to 0.

Since 8.1.400.37

- `n10` (**secondary notification**, by default 0) – If set to 1, then the agent ready event can trigger one extra VCB notification (if no call was routed to the agent). See **Step 6** in VCB Implementation.

Since 8.1.400.40

- `n11` (**dialing rate**, by default 150) – Defines *acceleration* factor of dial out rate. Value specified as percentage of *minimal* rate (one agent ready event triggers one notification). Valid range is 100-300. See **Step 7** in VCB Implementation.

Note: This option might conflict with the `DialOutSuccessRate` function. It is recommended not to mix them - in a scenario when dial out rate is controlled with the function for some calls and with the option for others, might result in a quite unstable dial out rate.

- `n12` (**over dial by waiting time**, by default 60) – Defines condition of over dial situation. If going through a queue of calls waiting for an agent, a former VCB call (that is, call already connected with the customer) will be detected as waiting for more than the specified time and URS will assume it has an over dial situation. If the value is set to 0, then detecting of over dial situations by time is disabled. See **Step 7** in VCB Implementation.

- `n13` (**over dial by calls**, by default 5) – Defines condition of over dial situation. If going through a queue of calls waiting an agent, the specified amount of former VCB calls (that is, calls already connected with the customer) will be detected and URS assumes it has an over dial situation. If the value is set to 0, then detecting of over dial situations by number of calls is disabled. See **Step 7** in VCB Implementation.

- `n14` (**over dial rate**, by default 0) – Defines dialing rate to be used if an over dial condition is detected. In an over dial situation, URS can totally suspend dialing (if option value is 0) or still consider making one VCB notification. The option defines the probability (in percentage) that the one notification will still be made. This option is intended to make the reaction to an over dial situation as efficient as possible, and reduce the rate of VCB notifications instead of suspending them completely. Valid values for this option are 0 to 100. See **Step 7** in VCB Implementation.

- `n15` (**reserving agent AHT**, by default 0) – slow queues case. If positive, it defines the average handing time threshold, on exceeding which URS in addition to sending a dial out notification will also block/reserve the agent after the VCB call. See **Step 7** in VCB Implementation.

- `n16` (**AHT gap history**, by default 0) – slow queues case. If positive, it defines the timeframe (in

seconds) within which URS will checks for gaps in handing time for the internal queue the VCB call is in. If a gap is detected, then URS, in addition to sending a dial out notification will also block/reserve the agent after the VCB call. See **Step 7** in VCB Implementation.

- n17 (**rest time**, by default 300) – used to extend call blocking time (n2) after sending a VCB notification. Normally it is expected that if dialing to customer fails (n2 expired) then the VCB application will place the VCB call into a route delay state for some extra time. In addition to this processing by the VCB application, an extra rest time for the call can be specified here. In effect, if sending a VCB notification results in no action, then the ext notification for this call will be sent not sooner than n2 + n17 seconds.

Note: Since 8.1.400.40, there is a possibility to specify the above listed VCB configuration parameters with separate options. The VCB options are specified in a dedicated `vcb` section. URS first checks the `default/vcb` option and if found sets the VCB parameters accordingly. In addition, URS will check the URS application for the `vcb` section and if found will try to set the VCB parameters with the specified values. The correct names of the options inside the section are as follows:

```
vcb_aht_threshold = n1
vcb_block_call = n2
vcb_block_agent = n3
vcb_multi_urs = n4
vcb_after_route_max = n5
vcb_max_notifications = n6
vcb_any_agent = n7
vcb_check_queue_max = n8
vcb_jump_priority = n9
vcb_secondary = n10
vcb_default_rate = n11
vcb_overdial_wait_time = n12
vcb_overdial_calls = n13
vcb_overdial_rate = n14
vcb_reserving_aht = n15
vcb_aht_gap_history = n16
vcb_rest_time = n17
```

Multiple URSes

Information related to using multiple URSes for VCB is provided in this topic.

Using multiple URSes is activated with the n4 `vcb` option. See VCB Configuration.

When using multiple URSes, agent reservation must be activated (even without the context of VCB). The `agent_reservation` URS option should be set to **true**, **generic** or **implicit** on all URSes. If needed, the associated URSes and T-Servers must also be configured to support the selected agent_reservation method.

Additionally (VCB specific):

- On all URSes (even those not involved in VCB but just routing calls to the same agents as the VCB URSes), the `vcb` option must have n4 set to 1 (see VCB Configuration).
- As a VCB notification cannot use implicit agent reservation, all URSes (even those not involved in VCB but just routing calls to the same agents as the VCB URSes) must be configured to be able to perform explicit agent reservation (regular or centralized). Even if the `agent_reservation` option is set to **implicit**, these URSes still need to be able to perform explicit agent reservation (have a connection to centralized or all involved T-Servers) if needed.

> ### Important
> Using an adjustable hit rate for VCB notifications has some limitations in case of multiple URSes. The multiple URSes case assumes agent reservation is in effect. Agent reservation request carries information only about one call (the one with the highest priority that URS node has). If more than 1 call was selected for notification (call1, call2, and so on), the positioning of the remaining calls in a *global* queue (containing calls from all URSes) is unknown. Other URSes might have routable calls of higher priority than call2, for example. This might result in the corresponding customer waiting for too long before being connected with the agent. A more advanced agent reservation mechanism is required to meet the requirement of this case in an appropriate way (each URS provides a small fragment of the calls from their local queues and gets back a fragment of calls in the global queue composed from pieces provided by each URS). Until such an advanced agent reservation mechanism is created it is recommended that, if using an adjustable hit rate for VCB notifications in a multiple URSes environment, the priority of every VCB call (after it is answered) must be increased in one way or another. (See the VCB Configuration section for more information on the `jump priority` setting).

Before 8.1.400.40, using of multiple URSes disables functionality described in Step 3 (Use Case B and C), VCB Implementation. Reasoning: URS does not know if some other URS was routing a call to the agent (Use Case B) and the content of the other URS's queue of calls. From 8.1.400.40, there is no such limitation, but the described limitations with using these functionalities in a multi-URS environment remains. As mentioned above, if activated (with parameters n8 and/or `n10` of the `vcb` option), it probably will be reasonable to increase the priority of every VCB call (after it is answered) in one way or another.

If multiple URSes are used for VCB but not configured as such (n4 in vcb is 0), the VCB functionality will still work. However, this can potentially result in over dialing of VCB calls (in efect, this is another way to increase the rate of dial notifications).

VCB Notification Structure

A VCB notifiable call must have in its extensions the `notifyurl` key. This key, together with a few other optional extensions, controls the location and format for sending VCB notifications.

The additional keys are:

- `notifybody`
- `notifyenc`.

The logic for sending a VCB notification is as follows:

For an ORS Instance

- If `notifyurl` starts with `ors://` then the entire `notifyurl` should be in the format, `ors://orsname/scxml/session/orssession/event/eventname[?params]`, and URS will try to send an event to the specified ORS and the specified session within it.
- If `notifyurl` starts with `ors:` then the entire `notifyurl` should be in the format, `ors:whatever/event/eventname[?params]`, and URS will try to send an event to the ORS node and the session associated with this VCB call.

In both the above cases, the optional `params` and `notifybody` can be provided.

- `params` should have the format of an URL encoded string of parameters.
- `notifybody` could either be an URL encoded string of parameters or a JSON string.

Their values if provided will be decoded first – any fragment in square brackets will be replaced with its actual value as given below:

[udata]	entire call attached data (& separated)
[udata.*]	entire call attached data (, separated)
[udata.key]	value of corresponding attached data
[udataj]	entire call attached data as JSON string
[ext]	all call extensions (& separated)
[ext.*]	all call extensions (, separated)
[ext.key]	value of corresponding extension key
[extj]	all call extensions as JSON string
[orssession]	ORS session ID if call has associated ORS session ID
[ors]	`host:port` of ORS node associated with this VCB call
[call.connid]	Connection ID of this VCB call

[call.uuid]	UUID of this VCB call

For an URS Instance

- If `notifyurl` starts with `urs://` then the entire `notifyurl` should be in the format, `urs://ursname/message` and URS will try to send a command to the specified URS.
- If `notifyurl` starts with `urs:` then the entire `notifyurl` should be in the format, `urs:message` and the particular URS will try to send a command to itself.

In both cases, the command is effectively invoking the `RequestRouter` function (updated description in the Supplement to the Universal Routing 8.1 Reference Manual):

```
RequestRouter[ursname, message, notifybody, ","]
```

Both `message` and `notifybody` preliminary will be decoded as described above.

For a Generic HTTP Server

In all other cases, URS will try to send a VCB notification as a REST HTTP message.

- `notifyurl` must be a valid URL.
- `notifybody` – If specified, URS will use POST message instead of GET message, and use `notifybody` as the content of this POST message.
- `notifyenc` – Used only if `notifybody` is present and will be used as the content of Content-Type header of the generated HTTP message.

Both `notifyurl` and `notifybody` preliminary will be decoded as described above.

VCB Sample

This section shows a sample VCB implementation in URS. It is simple and short using a completely URS based approach - all VCB components (sending and processing VCB notifications) are written as URS strategies.

To enable VCB functionality, the call processed by URS must be marked as VCB enabled. This is achieved by setting the EXECUTION_MODE key of the attached data to VCB. The attached data will not affect standard processing of inbound calls. However, if an inbound call happens to be abandoned before routing then URS will not drop it but will continue executing the strategy and will try to allocate the call to an available agent and connect the customer and the agent.

Just setting this attached data is enough to facilitate basic VCB of type 1 for any existing solution – no strategy changes or additional modifications are required – if (and when) needed, an outbound call to the customer will be generated and after answering, will be connected with the selected agent. This Type 1 VCB (when an agent is first selected/reserved after a call and an attempt to connect the customer and the agent is started) can be further adjusted if needed with extra attached data keys:

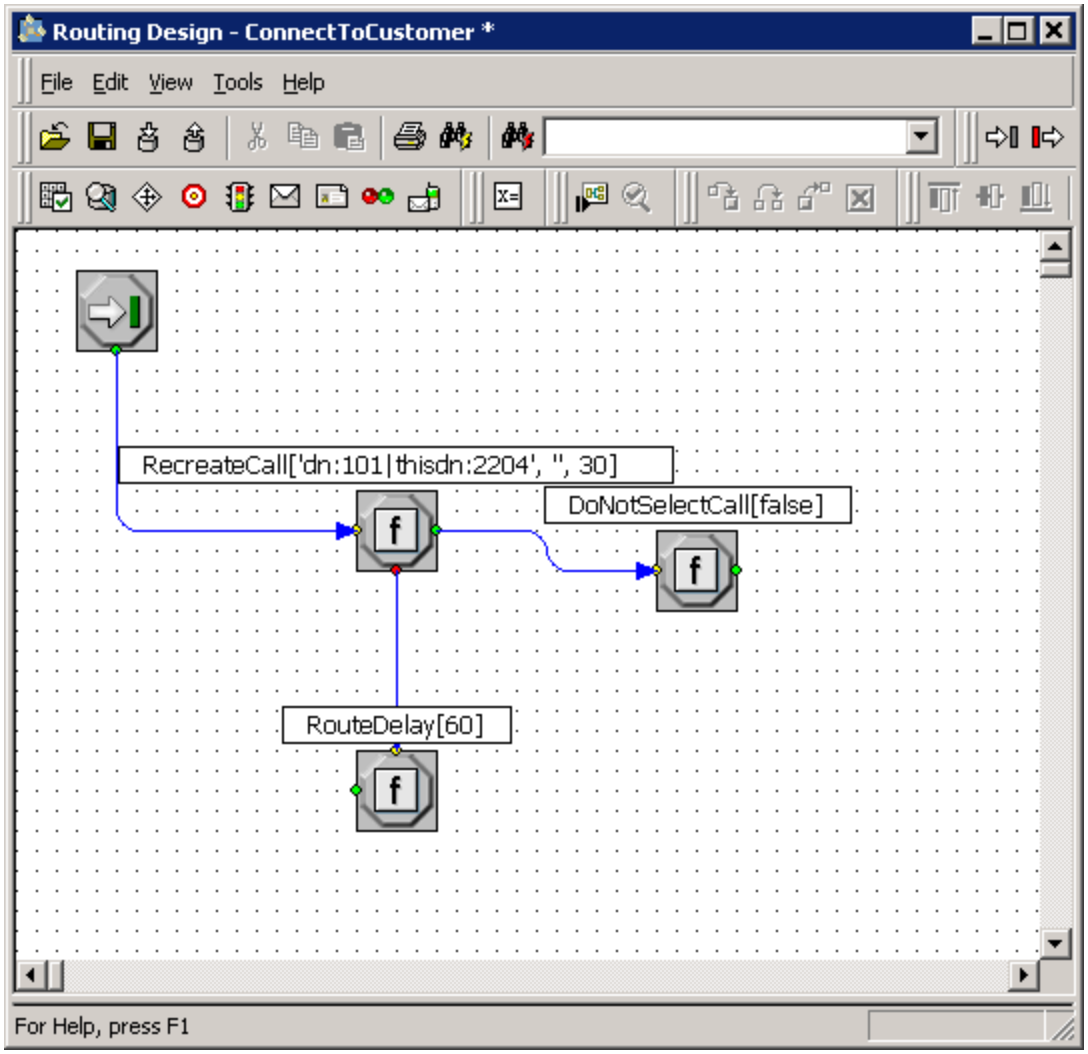
- `VCB_CONTACT` – where to dial customer (by default ANI is used)
- `VCB_STRATEGY` - if logic for processing inbound call after/if it was abandoned and turned into a VCB call must be different from the original logic and so on.

For implementing/enabling notification based VCB (VCB of type 2) a few additional steps must be performed (compared with the agent reserving VCB case).

- a) attached data EXECUTION_MODE=VCB - Common step required to make URS keep the call after the caller hangs up.
- b) execute function DoNotSelectCall[true] – To prevent URS from attempts to select an agent for this call (as agent is not supposed to be allocated until customer answers the outbound call).
- c) execute function ExtensionUpdate (for example, ExtensionUpdate['notifyurl', 'urs:urs/call/[call.connid]/start?strategy=ConnectToCustomer']) - Together with step b this will enable URS to activate logic in the `ConnectToCustomer` strategy when URS decides that it is time to dial the customer presented with this VCB call (name of the strategy can be anything; in this sample it is named `ConnectToCustomer`).

Note: The above will result in the following VCB notification to be distributed, `urs:urs/call/[call.connid]/start?strategy=ConnectToCustomer`. In other words, this means URS gets a *web* request to start executing the logic defined in the provided strategy in the context of the interaction with the provided `ConnID`. Execution of the logic is started in a parallel thread (see the description of the `start` URS WEB API method) and will not interfere with the main logic to be executed for this interaction.

- d) Write the above mentioned `ConnectToCustomer` strategy. The strategy is simple and can be directly encoded into `notifyurl` itself, for instance, by using the following `notifyurl` instead the one provided in step c: `urs:urs/call/[call.connid]/start?source=RecreateCall(ANI(), , 30); if(Failed()) RouteDelay(300); else DoNotSelectCall(false);`. Or this logic can be included into a separate strategy and referred to by name (see the `ConnectToCustomer` strategy below):

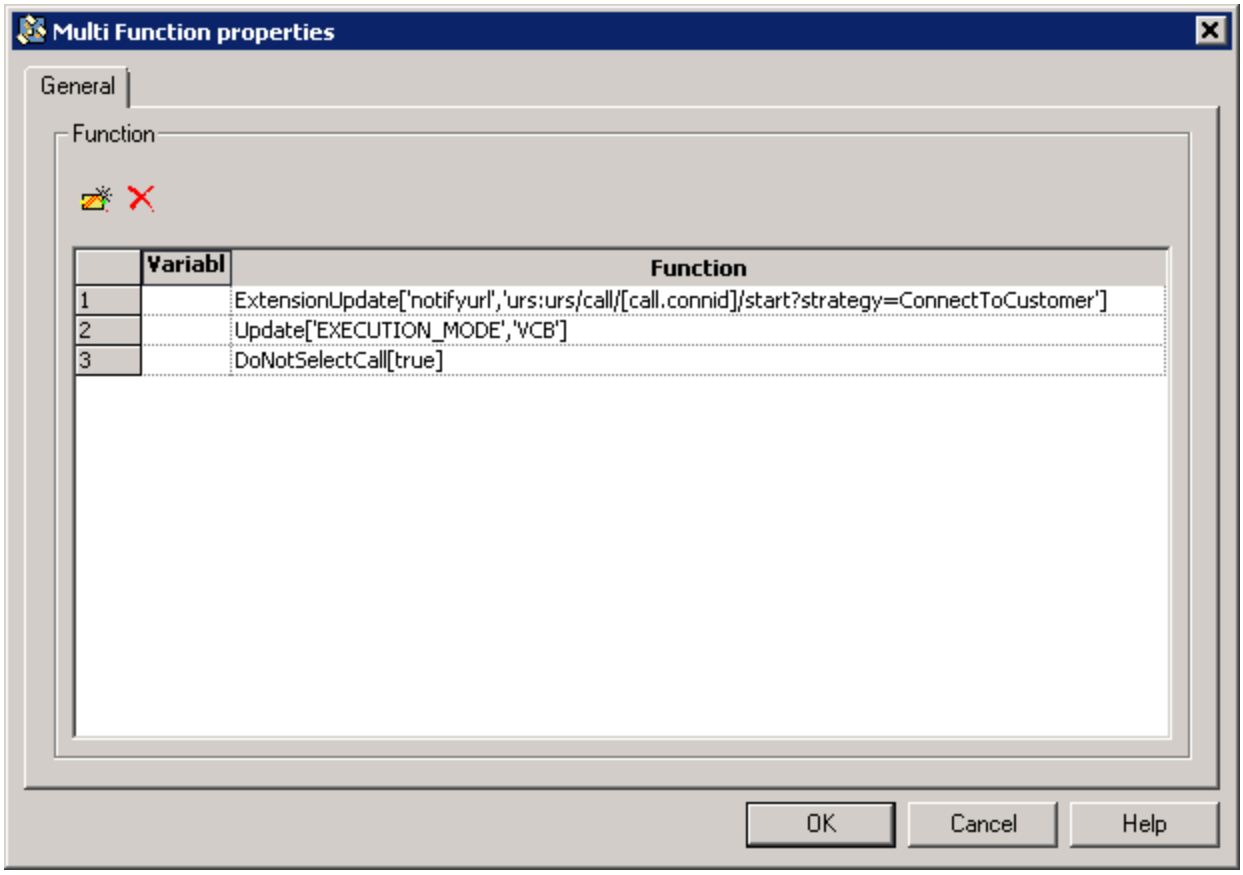


The `RecreateCall[dn, extensions, timeout]` function tries to establish connection with customer and if it succeeds, returns OK or otherwise an error (in effect, it converts a *virtual VCB call* into a real one). Its parameters are: where to dial the call, which extensions to specify in dialing request to T-Server, and how long to wait for customer to pick up the phone.

In case of success, the DoNotSelectCall flag is canceled and nothing prevents this call anymore from being routed to the next available agent. In case of failure, we pause for some time before the next attempt to dial the customer.

The `RouteDelay[time]` function will make a call *unnotifiable* (in addition to being *unroutable*) for the provided time and as such, URS will not try to issue another notification.

To summarize, activation of notification-based VCB consists of the following set of functions executed before the customer accepts the VCB offer:



Important

The `DoNotSelectCall[true]` function must be executed just before the customer accepts the VCB offer (and if he does not, then it should be canceled if it was already raised).

Restrictions, Disclaimer, and Copyright Notice